Projet PROGAV : ENSemenC

DEVOS Sacha

DELVAL Yannick

BORDEAUX INP Ensc

Sommaire

I. Introduction

Ce projet a été réalisé dans le cadre du semestre 6 ; l s'agit d'un simulateur de potager en C#, développé sous forme d'application console, avec pour objectif de mettre en œuvre les principes fondamentaux de la programmation orientée objet (POO).

Le jeu peut se dérouler dans un environnement virtuel représentant un pays fictif ou réel, influençant les types de semis disponibles et les conditions météorologiques. Le joueur est invité à gérer un potager au fil du temps, en effectuant différentes actions comme semer, soigner, ou récolter des plantes, tout en tenant compte de leurs préférences en matière de terrain, d'eau, de lumière, de température et de saison. Le jeu intègre également des éléments aléatoires, comme les intempéries ou les maladies, qui affectent la santé des plantes et obligent le joueur à s'adapter continuellement.

Outre les aspects ludiques, ce projet nous a permis de mobiliser un ensemble de compétences techniques : conception de classes abstraites, héritage, encapsulation, gestion des entrées utilisateur, affichage dynamique en console, respect des conventions de développement en C#... Nous avons également collaboré via GitHub tout au long du développement pour assurer une gestion efficace du code source et un partage fluide des tâches.

Le présent rapport technique a pour objectif de détailler l'univers du jeu, les règles de gameplay, les choix de conception orientée objet, les tests effectués pour valider le bon fonctionnement de l'application, ainsi qu'un bilan critique sur la conduite du projet.

II. Univers du jeu

Le jeu que nous avons développé est un simulateur de potager prenant place dans un univers réaliste inspiré du jardinage traditionnel, sans éléments imaginaires qui viendraient perturbé le côté réaliste que nous avons voulu donner à ce jeu. Il se déroule dans un environnement dynamique influencé par le climat, la saison, le type de terrain, et la région géographique choisie, ici la France.

1. Choix du pays

La partie se déroule donc en France, ce qui conditionne les plantes disponibles et les types de sol présents dans son potager. Ce choix impacte : la liste des semences autorisées (ex : tomates, carottes, salades…), les contraintes climatiques (météo réaliste par saison) ainsi que les possibilités de croissance des différentes espèces.

2. Types de terrain

Trois types de terrain sont générés et placés aléatoirement à l'initialisation du potager : Terre, Sable et Argile.

3. Les plantes

Le jeu comprend un ensemble varié de plantes qui poussent en France, en voici la liste : tournesol, cerisier, carotte, courgette, fraise, maïs, oignon, patate, rose, salade et tomate. Chaque plante possède : un cycle de vie spécifique (annuelle, vivace…), des conditions de plantation (saison, espacement, type de terrain…) et des besoins particuliers en eau, lumière, température, etc.

3.1 Les graines

Dans l’objectif de donner à l'utilisateur l’envie de cultiver correctement, on a ajouté des graines ; on en a 3 au départ et on peut en réussir qu’en faisant qu'une plante devienne mûre et la recueillir. Ceci force l’utilisateur à faire attention à ses choix ainsi qu’à prendre soin de la plante. Sinon, il se retrouvera sans graines !

3.2 Les fruits

Dans la même ligne on a les fruits, d'où sortent les graines.

4. Météo et saisons

Le jeu simule une météo dynamique par saison, influençant directement la santé des plantes : on modifie la température, les précipitations et l'ensoleillement, ainsi que les risques d'intempéries aléatoires tels que la grêle, le gel, la canicule…
De cette façon, les conditions climatiques peuvent soit accélérer la croissance si elles sont favorables, soit provoquer des maladies ou la mort prématurée si elles sont défavorables.
Afin de simuler les saisons de la façon la plus réaliste possible avons décidé que chaque saison devait durer 3 mois, ce qui permet de devoir se contraindre aux conditions particulières de chaque saison pendant un certain temps, sachant que chaque tour simule une semaine entière.

5. Présence d'événements dynamiques

Un point important du cahier des charges est la gestion d'événements inattendus comme la présence d'un rongeur ou des intempéries sévères (grêle, gel, etc.). Ces éléments sont désormais simulés en jeu. Par exemple, lorsqu'un rongeur apparaît, il peut voler des fruits. Le joueur a la possibilité de réagir rapidement pour limiter les pertes.De même, il y a des phénomènes météo extrêmes qui surviennent aléatoirement (grêle, canicule…), influençant la santé des plantes.
Ceci renforce d'avantage l'aspect réaliste que l'on a voulu donner à ce jeu

6. Cycle temporel et rythme du jeu

Le jeu adopte une double temporalité :

- ➔ Un mode classique (temps long), où chaque tour représente une semaine, avec progression régulière des plantes.

➔ Un mode urgence, déclenché aléatoirement en cas de menace (intempérie ou intrus), où l'affichage devient temporairement "instantané" pour simuler une réaction rapide.

III. Possibilités des joueurs et déroulement d'une partie

Le jeu propose une expérience interactive en console, centrée sur la gestion stratégique et progressive d'un potager. Le joueur dispose d'un ensemble d'actions qu'il peut réaliser à chaque tour, dans un environnement en constante évolution.

1. Actions possibles

À chaque tour (correspondant à une semaine de jeu), le joueur peut :

Avec les informations directement données sur la console :

- ➔ Observer l'état du potager : santé, position, maturité, terrain...
- ➔ Voir la météo et les informations environnementales.
- ➔ Agir en situation d'urgence, par exemple si un rongeur vole des fruits (mode urgence déclenché aléatoirement).

En interagissant avec le menu console :

- ➔ Planter une graine parmi les semences disponibles, en respectant les contraintes de saison, d'espacement, de type de sol, etc.
  Récolter les plantes mûres afin de récupérer des fruits et des graines pour les saisons suivantes.
- ➔ Soigner une plante malade ou blessée pour améliorer sa santé.
- ➔ Arroser une plante assoiffée si les précipitations ont été insuffisantes.
- ➔ Passer à la semaine suivante.
- ➔ Afficher les ressources en graines et en fruits.
- ➔ Afficher les types de sols disponibles dans son potager.
- ➔ Quitter le jeu.

2. Interface utilisateur

Toutes les actions sont accessibles via un menu console clair. Le potager est représenté visuellement grâce à une grille en émojis, chaque parcelle affichant une plante ou un type de sol. Une légende permet de comprendre instantanément la composition du jardin.

Exemple d'affichage : (CAPTURE D'CRAN)

3. Progression de la partie

Le déroulement du jeu suit un cycle temporel structuré :

- ➔ Chaque tour simule une semaine, au cours de laquelle les plantes poussent, tombent malades, prospèrent ou meurent en fonction des conditions (météo, terrain, soins...).
- ➔ Toutes les plantes sont évaluées automatiquement à la fin du tour.
- ➔ Des événements aléatoires (rongeurs ou intempéries) peuvent survenir et perturber le cours normal de la partie.
- ➔ Le joueur peut choisir de passer à la semaine suivante après avoir effectué les actions souhaitées.

4. Objectifs du joueur

Le joueur n'a pas de condition de victoire stricte, mais il est encouragé à :

- ➔ Faire pousser un maximum de plantes en bonne santé.
- ➔ Maintenir son potager productif et équilibré.
- ➔ Réagir efficacement aux événements imprévus.

IV. Choix de modélisation objet

Notre jeu repose sur une architecture orientée objet bien définie. Chaque composant clé du jeu (plante, météo, parcelle, potager, etc.) est modélisé sous forme de classes interconnectées. Voici un aperçu détaillé des principaux éléments et des choix de conception.

1. Structure générale des classes

- ➔ Plante (classe abstraite) : modèle les propriétés et comportements communs à toutes les plantes.
  - ◆ Quelques attributs : `Nom, Type, BesoinEau, BesoinLumiere, Espacement, Croissance`, etc.
  - ◆ Quelques méthodes : `Evaluer(), Soigner(), ProposerSoin(), Arroser(), ProposerArrosage()`.
- ➔ Classes filles de Plante : Tomate, Carotte, Rose, Cerisier, etc. qui définissent les valeurs spécifiques à chaque type.
- ➔ Meteo : gère les conditions climatiques dynamiques et leur évolution semaine après semaine.
  - ◆ Quelques attributs : `SaisonActuelle, Temperature, Precipitations, Ensoleillement`, etc.
  - ◆ Quelques méthodes : `GenererConditions(), IncrementerSemaine(, SaisonSuivante()`.
- ➔ Terrain et sous-classes (Terre, Sable, Argile) : représentent le type de sol d'une parcelle.
- ➔ Parcelle : contient une Plante (ou rien) et un Terrain.
- ➔ Potager : contient une grille 2D de Parcelle, et centralise les interactions sur les plantes.
  - ◆ Quelques attributs : `Largeur, Hauteur, Parcelle[,] Grille`.
  - ◆ Quelques méthodes : `Planter(), EvaluerPlantes(), AfficherEtat(), AfficherGrille(), Recolter()`, etc.
- ➔ Program.cs : point d'entrée du jeu, contient la boucle principale, le menu, et la logique d'interaction.

2. Relations entre les classes

➔ Composition :

- ◆ Potager ↔ Parcelle : Potager contient une matrice de Parcelle.
- ◆ Parcelle ↔ Plante et Parcelle ↔ Terrain : Chaque parcelle contient une plante et un sol.

➔ Héritage :

- ◆ Plante ← Tomate, Carotte, Fraise … : Chaque sous-classe hérite des attributs communs de Plante et initialise ses propres valeurs dans le constructeur.
- ◆ Terrain ← Terre, Sable, Argile : Ces classes représentent les différents types de sol et héritent d'un attribut commun TypeSol.

➔ Agrégation :

- ◆ Potager → Graines, Fruits : utilisées pour gérer les stocks.

3. Justification des choix

➔ La séparation des responsabilités est claire : la météo est indépendante des plantes, et chaque plante connaît son état.

➔ L'utilisation de l'héritage pour Plante permet de factoriser le comportement commun tout en autorisant la personnalisation par espèce.

➔ Le potager manipule la grille sans avoir besoin de connaître les détails internes de chaque plante.

➔ L'évaluation repose sur des interactions naturelles entre objets comme (Plante.Evaluer(Meteo, Terrain)).

V. Tests de fonctionnement

Des tests ont été effectués tout le long du déroulement du jeu, permettant de s’assurer que le jeu fonctionne correctement par rapport à ce qui est attendu et qu’il n’y a pas de bugs qui empêchent l’utilisation du jeu.

Tout d'abord, la phase de plantation a été soumise à des tests concluants : les semis ne peuvent être réalisés que durant la saison adéquate, sur le type de sol requis, et dans un espace suffisant. Le jeu restreint les actions invalides tout en fournissant des indications claires au joueur via des messages explicites.

Ensuite, la phase de croissance des plantes a été évaluée chaque semaine. Des variations de santé ont été observées en fonction des conditions météorologiques (ensoleillement, pluie, température), et des événements spéciaux (grêle, canicule) se produisent de manière réaliste. L'arrosage a été déclenché en cas de manque d'eau de la plante et celle-ci obtient bien un bonus de santé.

La récolte fonctionne correctement : seules les plantes arrivées à maturité peuvent être récoltées, générant ainsi des fruits et des graines. Tenter de récolter trop tôt est bien bloqué.

Le mode d'urgence a été testé dans les deux cas possibles. Les alertes s'affichent correctement, laissant à l'utilisateur le choix d'intervenir. Le rongeur est toujours susceptible de dérober des fruits, mais en moindre quantité si le joueur réagit vite.

Enfin, l'interface graphique a été vérifiée : les plantes sont facilement discernables grâce à des emojis, les coordonnées sont toujours mentionnées, et les erreurs de saisie (coordonnées incorrectes, plantes inconnues...) sont correctement gérées

VI. Gestion de projet

Pour ce projet, voici l'organisation qui a été choisie afin d'être le plus efficace possible et de se répartir au mieux les tâches.

1. Rôles et tâches

Yannick Delval :

- ➔ Création de l'environnement gitignore
- ➔ Réalisation de diverses fonctions (spécifié dans la matrice d'implication).
- ➔ Gestion des corrections de bugs identifiés durant les tests.
- ➔ Rédaction du justificatif technique et de la matrice.

Sacha Devos :

- ➔ Réalisation de diverses fonctions (spécifié dans la matrice d'implication).
- ➔ Gestion des corrections de bugs identifiés durant les tests.
- ➔ Rédaction du justificatif technique et de la matrice.

2. Méthodologie

Le travail a été réalisé en binôme sur une période de 8 semaines. Nous avons utilisé une approche collaborative par partage de tâches selon les disponibilités. Aucune séparation rigide n'a été imposée : chaque membre pouvait intervenir sur toutes les parties du code.

Des points de synchronisation réguliers (plusieurs fois par semaine) ont permis de :

- ➔ Relire et tester les modifications apportées par l'autre.
- ➔ Identifier les bugs ou incohérences et les corriger rapidement.

Nous avons également veillé à équilibrer les tâches complexes avec des tâches plus simples pour maintenir une répartition équitable.

3. Planning de réalisation

Étape	Délai (semaines)	Tâches
Initialisation	1	Analyse des consignes, choix des étapes à suivre et répartition des tâches.
Développement	6	Implémentation des fonctionnalités.
Tests	Tout le long du développement	Test du jeu, identification et correction des bugs.
Finalisation	1	Documentation et dépôt final sur GitHub.

VII. Bilan critique et conclusion

Ce projet de simulation de potager a été particulièrement enrichissant à plusieurs niveaux.

Points positifs :

- ➔ Approfondissement de la programmation orientée objet : Nous avons pu appliquer concrètement des notions comme l'héritage, l'encapsulation, ou encore les interactions entre classes, avec une structure bien segmentée.
- ➔ Travail en binôme efficace : La coopération constante, le partage des tâches et les échanges réguliers ont permis d'éviter les blocages et de progresser de façon fluide tout au long des 8 semaines.
- ➔ Cela a été satisfaisant de voir un jeu se concrétiser par notre effort.

Limites et difficultés :

- ➔ Complexité croissante : Le nombre de fonctionnalités ayant augmenté, certaines parties du code sont devenues denses, ce qui a parfois ralenti les phases de tests ou de corrections.
- ➔ Affichage console limité : Bien que nous ayons fait au mieux pour rendre l'interface lisible (emojis, coordonnées, couleurs...), l'affichage reste parfois difficile à suivre, surtout quand il y a beaucoup de plantes.
- ➔ Temps restreint pour les bonus : Certaines idées intéressantes (comme le magasin ou une IA de suggestion) n'ont pas pu être entièrement développées par manque de temps.

Axes d'amélioration :

- ➔ Repenser l'affichage pour le rendre plus clair et ergonomique.
- ➔ Réorganiser certaines classes pour encore mieux séparer logique métier et affichage.
- ➔ Appliquer des fonctionnalités bonus telles que le magasin, qui offre une dimension économique au jeu et pas uniquement agricultrice.

En conclusion, ce projet nous a permis de mieux maîtriser la programmation orientée objet en C#, de travailler efficacement en équipe, et de produire une application cohérente et ludique, malgré quelques compromis liés au temps imparti.